Лабораторная работа №3

Вариант 10.

Задание 1. В соответствии с заданием своего варианта составить программу для вычисления значения функции с помощью разложения функции в степенной ряд. Задать точность вычислений eps.

10. $$\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n = 1 + x + x^2 + \ldots, |x| < 1$$

```
1 usage
def calculate_function(x, eps):
    """Calculate the sum of the series 1 + x + x^2 + x^3 + ... with given accuracy eps."""

    term = 1
    sum = 1
    n = 0

    while abs(term) > eps and n < 500:
        term *= x
        sum += term
        n += 1

    return sum, n

1 usage
def input_values():
    """Input x and eps values from user and validate them."""

    while True:
        try:
            x = float(input("Введите значение x такое, что (|x| < 1): "))
            if abs(x) >= 1:
                raise ValueError("Введенное значение должно быть меньше 1 по модулю")
            break
        except ValueError as e:
            if isinstance(e, ValueError) and "could not convert string to float" in str(e):
                print("Ошибка: Введено не числовое значение. Попробуйте еще раз.")
            else:
                print(f"Ошибка: {e}. Попробуйте еще раз.")
```

calculate_function()

```
                print("Ошибка: Введено не числовое значение. Попробуйте еще раз.")
            else:
                print(f"Ошибка: {e}. Попробуйте еще раз.")
    while True:
        try:
            eps = float(input("Введите точность вычислений eps: "))
            break
        except ValueError:
            print("Ошибка: Введено не числовое значение. Попробуйте еще раз.")
    return x, eps

1 usage
def output_values(x, sum, n, eps):
    """Output x, n, sum, and math value of the function 1/(1-x) and eps to the console."""

    math_value = 1/(1-x)
    print(f"x : {x}")
    print(f"n : {n}")
    print(f"F(x) : {sum}")
    print(f"Math F(x) : {math_value}")
    print(f"eps : {eps}")

2 usages
def task1_solve():
    """Solve task 1."""

    x, eps = input_values()
    sum, n = calculate_function(x, eps)
    output_values(x, sum, n, eps)
```

```
Введите номер задачи (0, чтобы завершить программу) : 1
Введите значение x такое, что (|x| < 1): 0.5
Введите точность вычислений eps: 0.0001
x : 0.5
n : 14
F(x) : 1.99993896484375
Math F(x) : 2.0
eps : 0.0001

Введите номер задачи (0, чтобы завершить программу) :
```

Задание 2. В соответствии с заданием своего варианта составить программу для нахождения суммы последовательности чисел.

10.	Организовать цикл, который принимает целые числа и вычисляет наибольшее из них. Окончание цикла – ввод числа 0

```python
def input_values():
    '''Input integer values from user and validate them'''
    num_list = []
    while True:
        try:
            num = int(input("Введите целое число (0, чтобы закончить ввод) : "))
            if num == 0:
                break
            num_list.append(num)
        except ValueError:
            print("Ошибка. Необходимо целочисленное значение. Попробуйте еще раз")
    return num_list

def calculate_max(num_list):
    '''Calculate the maximum number in the list'''
    max_num = max(num_list)
    return max_num

def calculate_sum(num_list):
    """Calculate the sum of the numbers in the list"""

    sum_num = sum(num_list)
    return sum_num
```

```python
def task2_solve():
    """Solve task 2"""

    num_list = input_values()
    print(num_list)
    max_num = calculate_max(num_list)
    print(f"Максимальное число: {max_num}")
    sum_num = calculate_sum(num_list)
    print(f"Сумма чисел: {sum_num}")
```

```
Введите номер задачи (0, чтобы завершить программу) : 2
Введите целое число (0, чтобы закончить ввод) : 2
Введите целое число (0, чтобы закончить ввод) : 3
Введите целое число (0, чтобы закончить ввод) : 1
Введите целое число (0, чтобы закончить ввод) : 8
Введите целое число (0, чтобы закончить ввод) : 0
[2, 3, 1, 8]
Максимальное число: 8
Сумма чисел: 14

Введите номер задачи (0, чтобы завершить программу) :
```

Задание 3.

10.	В строке, вводимой с клавиатуры, подсчитать количество символов, лежащих в диапазоне от 'g' до 'o'

```python
1 usage
def count_all_chars_decorator(func):
    """Decorator that counts the total number of characters in the text."""

    def wrapper(text):
        total_chars = sum(char.isalpha() for char in text)
        print(f"Общее число букв: {total_chars}")
        return func(text)
    return wrapper


1 usage
@count_all_chars_decorator
def count_chars(text):
    """Returns the number of characters 'g' to 'o' in the text."""

    count = 0
    for char in text:
        if 'g' <= char <= 'o':
            count += 1
    return count


2 usages
def task3_solve():
    """Solve task 3"""

    text = input("Введите текст: ")
    count = count_chars(text)
    print(f"Количество символов от 'g' до 'o': {count}")
```

wrapper вызывается вместо оригинальной

Функция wrapper сначала вычисляет общее количество буквенных символов в тексте, используя генератор списка и встроенную функцию sum. Затем она выводит это

количество на экран. После этого функция wrapper вызывает оригинальную функцию func с аргументом text и возвращает ее результат.

```
Введите номер задачи (0, чтобы завершить программу) : 3
Введите текст: anc beo nsh
Общее число букв: 9
Количество символов от 'g' до 'o': 4
```

Задание 4.

10.	а) определить число слов, ограниченных пробелами; б) определить, сколько раз повторяется каждая буква; в) вывести по алфавиту словосочетания, отделенные запятыми

```
1 usage
def word_count(text):
    """Returns the number of words in the text."""

    words = text.split()
    return len(words)


1 usage
def letter_repetition(text):
    """Returns the number of repetitions of each letter in the text."""

    letter_counts = {}
    for letter in text.lower():
        if letter.isalpha():
            if letter in letter_counts:
                letter_counts[letter] += 1
            else:
                letter_counts[letter] = 1
    return letter_counts


1 usage
def alphabetical_phrases(text):
    """Returns the phrases separated by commas in alphabetical order."""

    phrases = [phrase.strip() for phrase in text.lower().split(',')]
    phrases.sort()
    return phrases
```

```python
def task4_solve():
    """Solve task 4"""

    text = "So she was considering in her own mind, as well as she could, for the hot day made her feel very sleepy and st
    print(f"Анализируемый текст:\n{text}")

    count = word_count(text)
    print(f"\nа) число слов, ограниченных пробелами: {count}")

    print("\nб) количество повторений каждой буквы:")
    letter_counts = letter_repetition(text)
    print(sorted(letter_counts.items()))


    phrases = alphabetical_phrases(text)
    print("\nв) словосочетания, отделенные запятыми, в алфавитном порядке:\n")
    for phrase in phrases:
        print(phrase)

```

```
Введите номер задачи (0, чтобы завершить программу) : 4
Анализируемый текст:
So she was considering in her own mind, as well as she could, for the hot day made her feel very sleepy and stupid, whether the pleasure of making a daisy-chain would be wo

а) число слов, ограниченных пробелами: 55

б) количество повторений каждой буквы:
[('a', 16), ('b', 5), ('c', 5), ('d', 13), ('e', 31), ('f', 4), ('g', 5), ('h', 17), ('i', 17), ('k', 3), ('l', 10), ('m', 3), ('n', 15), ('o', 12), ('p', 6), ('r', 12), ('

в) словосочетания, отделенные запятыми, в алфавитном порядке:

as well as she could
for the hot day made her feel very sleepy and stupid
so she was considering in her own mind
when suddenly a white rabbit with pink eyes ran close by her.
whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies
```

Задание 5.

Задание 5. В соответствии с заданием своего варианта составить программу для обработки вещественных списков. Программа должна содержать следующие базовые функции:

1) ввод элементов списка пользователем;

2) проверка корректности вводимых данных;

3) реализация основного задания с выводом результатов;

4) вывод списка на экран.

	элементов списка, расположенных до максимального элемента
10.	Найти минимальный положительный элемент списка и сумму элементов списка, расположенных между первым и последним положительными элементами

```python
# list_initialization.py

1 usage
def input_list(lst, n):
    """Function to enter a list of n real numbers."""

    i = 0
    while i < n:
        num = input("Введите вещественное число: ")
        if validate_input(num):
            lst.append(float(num))
            i += 1
        else:
            print("Введено некорректное значение. Попробуйте еще раз.")
    return lst

1 usage
def float_generator(n):
    """Generator function to generate n real numbers."""

    for i in range(n):
        yield float(i) + 1.25
1 usage
def input_list_generator(lst, n):
    """Function to enter a list of n real numbers using a generator."""

    for num in float_generator(n):
        lst.append(num)

```

```python
1 usage
def validate_input(input_value):
    """Function to validate input value."""

    try:
        float(input_value)
        return True
    except ValueError:
        return False
```

```python
import list_initialization

1 usage
def process_list(lst):
    """Return the minimum positive number and the sum of the numbers between the first and last."""

    positive_nums = [num for num in lst if num > 0]
    if not positive_nums:
        return None, None
    min_positive = min(positive_nums)
    first_positive_index = next(i for i, num in enumerate(lst) if num > 0)
    last_positive_index = len(lst) - 1 - next(i for i, num in enumerate(lst[::-1]) if num > 0)
    sum_between = sum(lst[first_positive_index + 1:last_positive_index])
    return min_positive, sum_between


1 usage
def print_list(lst):
    """Print the list."""

    print("Список:", lst)

```

```python
def input_list(lst):
    """Input a list of n real numbers."""

    print("\nВыберите способ создания списка\n1 - С помощью пользовательского ввода\n2 - С помощью функции генератора\n")
    while True:
        try:
            sub_task_num = int(input("Введите значение для выбора: "))
            if sub_task_num != 1 and sub_task_num != 2:
                raise ValueError("Введено некорректное значение. Введите 1 или 2.")
            break
        except ValueError as e:
            if "invalid literal for int() with base 10" in str(e):
                print("Ошибка: Введено не числовое значение. Попробуйте еще раз.")
            else:
                print(f"Ошибка: {e}. Попробуйте еще раз.")

    while True:
        try:
            n = int(input("Введите размерность списка: "))
            if n <= 0:
                raise ValueError("Размерность списка должна быть положительным числом.")
            break
        except ValueError as e:
            if "invalid literal for int() with base 10" in str(e):
                print("Ошибка: Введено не числовое значение. Попробуйте еще раз.")
            else:
                print(f"Ошибка: {e}. Попробуйте еще раз.")
```

```
    if sub_task_num == 1:
        list_initialization.input_list(lst, n)
    elif sub_task_num == 2:
        list_initialization.input_list_generator(lst, n)


2 usages
def task5_solve():
    """Solve task 5."""

    lst = []
    input_list(lst)

    while True:
        print("\n1 - Ввести новый список\n2 - Вывести список\n3 - Найти минимальный положительный элемент списка и "
              "сумму элементов списка, расположенных между первым и последним положительными элементами\n0 - "
              "завершить задание\n")
        while True:
            try:
                task_num = int(input("Введите номер функции : "))
                if task_num < 0 or task_num > 3:
                    raise ValueError("Введенное значение должно быть в диапазоне от 0 до 3.")
                break
            except ValueError as e:
                if "invalid literal for int() with base 10" in str(e):
                    print("Ошибка: Введено не числовое значение. Попробуйте еще раз.")
                else:
                    print(f"Ошибка: {e}. Попробуйте еще раз.")
```

```
                    print(f"Ошибка: {e}. Попробуйте еще раз.")
        if task_num == 1:
            lst.clear()
            input_list(lst)

        elif task_num == 2:
            print_list(lst)
        elif task_num == 3:
            min_positive, sum_between = process_list(lst)
            if min_positive is None:
                print("В списке нет положительных чисел.")
            else:
                print(f"Минимальное положительное число: {min_positive}")
                print(f"Сумма чисел между первым и последним положительными числами: {sum_between}")
        elif task_num == 0:
            break

```

```
Введите номер задачи (0, чтобы завершить программу) : 5

Выберите способ создания списка
1 - С помощью пользовательского ввода
2 - С помощью функции генератора

Введите значение для выбора: 2
Введите размерность списка: 4

1 - Ввести новый список
2 - Вывести список
3 - Найти минимальный положительный элемент списка и сумму элементов списка, расположенных между первым и последним положительными элементами
0 - завершить задание

Введите номер функции : 2
Список: [1.25, 2.25, 3.25, 4.25]

1 - Ввести новый список
2 - Вывести список
3 - Найти минимальный положительный элемент списка и сумму элементов списка, расположенных между первым и последним положительными элементами
0 - завершить задание
```

```
Введите номер функции : 1

Выберите способ создания списка
1 - С помощью пользовательского ввода
2 - С помощью функции генератора

Введите значение для выбора: 1
Введите размерность списка: 3
Введите вещественное число: 1.2
Введите вещественное число: 1.4
Введите вещественное число: 5

1 - Ввести новый список
2 - Вывести список
3 - Найти минимальный положительный элемент списка и сумму элементов списка, расположенных между первым и последним положительными элементами
0 - завершить задание

Введите номер функции : 3
Минимальное положительное число: 1.2
Сумма чисел между первым и последним положительными числами: 1.4
```